Algorithms **2014**, *7*, 663-684; doi:10.3390/a7040663

OPEN ACCESS

algorithms

ISSN 1999-4893

www.mdpi.com/journal/algorithms

Article

COOBBO: A Novel Opposition-Based Soft Computing Algorithm for TSP Problems

Qingzheng Xu [1,*], Lemeng Guo [1], Na Wang [2] and Yongjian He [1]

[1] Department of Information Service, Xi'an Communications Institute, No. 8, Zhangba East Road, Xi'an 710106, China; E-Mails: glmdick@163.com (L.G.); heshifu@sina.com (Y.H.)

[2] Department of Basic Courses, Xi'an Communications Institute, No. 8, Zhangba East Road, Xi'an 710106, China; E-Mail: syesun@hotmail.com

* Author to whom correspondence should be addressed; E-Mail: xuqingzheng@hotmail.com; Tel.: +86-29-8470-6536.

External editor: Toly Chen

Received: 9 October 2014; in revised form: 26 November 2014 / Accepted: 8 December 2014 / Published: 12 December 2014

Abstract: In this paper, we propose a novel definition of opposite path. Its core feature is that the sequence of candidate paths and the distances between adjacent nodes in the tour are considered simultaneously. In a sense, the candidate path and its corresponding opposite path have the same (or similar at least) distance to the optimal path in the current population. Based on an accepted framework for employing opposition-based learning, Oppositional Biogeography-Based Optimization using the Current Optimum, called COOBBO algorithm, is introduced to solve traveling salesman problems. We demonstrate its performance on eight benchmark problems and compare it with other optimization algorithms. Simulation results illustrate that the excellent performance of our proposed algorithm is attributed to the distinct definition of opposite path. In addition, its great strength lies in exploitation for enhancing the solution accuracy, not exploration for improving the population diversity. Finally, by comparing different version of COOBBO, another conclusion is that each successful opposition-based soft computing algorithm needs to adjust and remain a good balance between backward adjacent node and forward adjacent node.

Keywords: biogeography-based optimization; opposition-based learning; traveling salesman problems; discrete domain; opposite path; population diversity

---

1. Introduction

An opposition concept is both familiar and mysterious at the same time to ordinary mortals like us. It is familiar in that it is unconsciously or consciously applied in our regular life. That said, the opposition concept is also mysterious, in that it has different meaning and name for different academic disciplines. Over the last 2500 years, the study of opposites has attracted the attention of countless experts and scholars in various fields. Perhaps it is only because opposite terms and the nature of opposites intrigue and fascinate us. However, due to the lack of an accepted mathematical or computational model, until recently it has not been explicitly investigated to any great length in the fields outside of philosophy and logic [1].

Obviously, this awkward situation has changed in the last decade under the unremitting efforts of computer scientists. The basic concept of Opposition-Based Learning (OBL) was original introduced in 2005 by Tizhoosh [2]. Soon after, some new varieties of opposition-based learning were proposed for use: Quasi-Opposition-Based Learning (QOBL) [3], Quasi-Reflection Opposition-Based Learning (QROBL) [4], Center-based Sampling [5], Generalized Opposition-Based Learning (GOBL) [6] and Opposition-Based Learning using the Current Optimum (COOBL) [7]. The essential idea of these optimization strategies is the serious consideration of an estimate and its corresponding opposite estimate simultaneously to achieve an optimal approximation of the current candidate solutions. Nevertheless, a major distinction between these technologies lies in how to define the concept of opposite estimate as a part of enhancing population diversity. From the perspective of algorithm design, mathematical and experimental comparison between opposition-based sampling and random sampling is one of the most widely used techniques for continuous space. Recently, some mathematical proofs were explored independently in order to show why the opposition-based learning is beneficial when no preliminary knowledge about the solution is available [8–10].

It is amazing that they have been utilized in a vast majority of soft computing areas over a very short period of time. These simple and efficient meta-heuristic methods mainly include Differential Evolution (DE) [3,6,7,11,12], Particle Swarm Optimization (PSO) [13–15], Reinforcement Learning (RL) [2,16], Biogeography-Based Optimization (BBO) [4,17], Artificial Neural Network (ANN) [18,19], Harmony Search (HS) [20,21], Ant Colony System (ACS) [22,23] and Artificial Bee Colony (ABC) [24,25]. At present, the most successful applications of the ideas of OBL and its variants focus on traditional optimization fields, such as large-scale unconstrained optimization problem, constrained optimization problem, multi-objective optimization problem, and optimization problem in noisy environment. For a detailed overview of a range of opposition-based learning and its typical applications, we recommend a recent review article by Xu, *etc.*, entitled "A review of opposition-based learning from 2005 to 2012" [26].

Unfortunately, until now, opposition-based soft computing algorithms are rarely designed and applied to classical discrete optimization problems such as the minimum spanning tree problem, knapsack problem or scheduling problem. As far as we know, [22,27,28] are the few published papers for solving successfully the Traveling Salesman Problem (TSP), a famous combinatorial problem. The goal of this paper is to define a novel opposite numbers in discrete domain and then compare it with other acknowledged definitions of opposite solution. Furthermore, what deserves special note here is that, although the proposed scheme is embedded in the classical Biogeography-Based Optimization

(BBO) algorithm in this paper, in practice it is general enough to be applied to almost all soft computing algorithms to improve its performance. In a way, this paper is only the beginning of what you can do with the novel opposite numbers in discrete domain.

The general thinking of BBO, TSP and the well-known definition of OBL in continuous and discrete domain are introduced briefly in Section 2. In Section 3, an optimization method, Oppositional Biogeography-Based Optimization using the Current Optimum (COOBBO), for TSP problems, is proposed based on a new definition of opposite path in discrete domain. The simulation results comparing COOBBO with some other optimization methods and detailed discussion are given in Section 4. Finally, some concluding remarks and suggestions for further research are presented in Section 5.

It is important to mention that this work is a revised and expanded version of a paper entitled "A novel oppositional biogeography-based optimization for combinatorial problems" presented at the 2014 10th International Conference on Natural Computation (ICNC 2014), Xiamen, China, 19–21 August 2014 [29]. The present paper extends the previous research work by the following important contributions. (1) At the beginning of Section 3, the full details and motivation of proposed scheme is clearly stated to make it easier to understand. (2) The other two definitions of opposite path are proposed in the similar way as the previous one in [29]. Then the three definitions of opposite path are compared carefully in order to show their own advantages and disadvantages, and further to sum up experience conscientiously about new idea and design of opposite path. (3) In the part of results and discussions, a new comparison criteria, population diversity, is introduced and then it is analyzed for all tested algorithms. Later, a random version of OBBO is also introduced and compared with original OBBO algorithm in this paper. In addition, all experimental results of computation time are fitted, and then relative fitting errors are computed to support our previous guess in [29].

2. Background

2.1. Biogeography-Based Optimization

As a relatively young evolutionary algorithm, Biogeography-Based Optimization was first introduced in 2008 by Simon [30]. It is inspired by the science of biogeography which studies the distribution of species and ecosystems amongst islands in geographic space and through geological time.

In BBO, a population of candidate solutions is generated randomly, each candidate representing an island. These islands are then assigned, based on their fitness, immigration and emigration rates respectively, which are the important features of BBO that distinguish it from other evolutionary algorithms. The emigration rate indicates how likely a solution is to share its features with other solutions, and the immigration rate indicates how likely a solution is to accept features from other solutions. For example, a candidate solution with higher fitness for optimization problem will have a higher emigration rate and lower immigration rate, so that it can share its features with less fit islands and it is less likely to be spoiled by migrations from less fit islands. In contrast, a worse solution candidate will be assigned a high immigration rate and accept a lot of new features from relative good solutions. This addition of new features to the poor solution will, inevitably, enrich the fitness of the island. Throughout a lot of iterations, biological species migrate among different islands based on the immigration and emigration rates to find a better habitat. For further details please carefully read the reference cited at the beginning of the subsection.

2.2. Traveling Salesman Problems

It is well known that the combinatorial problems are not new to heuristic algorithms and have fascinated many scientists and engineers over the past few decades. As a matter of fact, they are considered as standard benchmarks for heuristic algorithms. For example, TSP is a well-known and ancient combinatorial problem since being defined in the 1800s by the Irish mathematician Hamilton and by the British mathematician Kirkman [31]. To summarize, there are three major reasons that the TSP has become a standard benchmark for soft computing algorithms [32]. First, the TSP is very intuitive and quite easy to state in terms of the complete graph on N vertices, and it is similar to many practical problems such as automatic assembly and configuration planning, sensor selection and power allocation over wireless networks, robotic path planning, and many others. Second, the TSP can easily be modified to become a multi-objective problem and solving multi-objective problems is a practical challenge but perhaps not insurmountable in many areas of engineering and industry. Third, an optimal TSP solution is extremely hard to obtain by using analytical methods. Even using numerical methods, it is still quite a challenge on a large number of real problem instances. The running time for brute-force search approach lies within a polynomial factor of O(n!). For instance, for a problem with 20 cities by brute force searching, it would be (20 − 1)!/2 possible tours, so this solution becomes impractical within limited computing time and memory space.

Because of the reasons above, many TSP problems with different diversities (even over 80,000) are formed to challenge the performance of existing heuristic algorithms, and many new algorithms inspired by nature and biological processes are specially proposed to conquer the TSP more efficiently. It is believed that our work in this paper is a useful attempt following this way, though we have not yet found the optimal solutions for symmetric TSP.

2.2. Oppositional Biogeography-Based Optimization for Combinatorial Problems

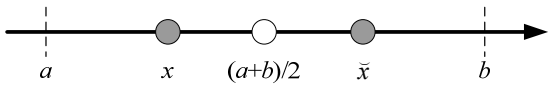
When dealing with D dimensional vectors in continuous domain (the space would be $\mathbf{R}^D$), the definition of opposite numbers is first given by Tizhoosh as follows [2].

Definition 1. Let $P = (x_1, x_2, ..., x_D)$ be a point in D-dimensional space, where $x_1, x_2, \ldots, x_D \in \mathbf{R}$ and $x_i \in [a_i, b_i]$, $\forall\ i \in \{1, 2, \ldots, D\}$. The opposite point $\breve{P} = (\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ is completely defined by its coordinates

$$\breve{x}_i = a_i + b_i - x_i \tag{1}$$

Figure 1 illustrates the opposite point $\breve{P}$ in one dimensional case.

Figure 1. Opposite point defined in domain [a, b]. x is a candidate solution and $\breve{x}$ is the opposite of x.



To the best of our knowledge, the great majority of published research papers on opposition-based learning are for solving continuous domain optimization problems. Recently, more and more researchers gradually realized that opposition-based learning, originally introduced for accelerating a continuous search space, can also be modified to be used alongside BBO to solve combinatorial problems, including TSP.

Obviously, wherein the main difficulty of these algorithm extensions is that how to define and evaluate opposite numbers in discrete domain [26]. For instance, given a candidate path, the attempting to apply opposition concept by simply reversing its order of the nodes is meaningless because the reversed path will yield the same cost as the original path in a TSP problem. For example, a tour (1, 2, 3, 4, 1) and its opposite tour, (1, 4, 3, 2, 1), have the same cost values because all of the cities preserve their neighbors. As a result, based on superficial analysis above, a new definition of opposition for TSP is needed and it should be different appreciably from that for continuous domain problems. In [28], an opposite path as following was defined as a candidate path that maximizes the distance between the adjacent vertices in the original path.

Definition 2. Let n be the number of nodes in a graph and $P = [1, 2, \ldots, n]$ be an even node cycle. The clockwise opposite path, P_o^{CW}, is defined as

$$P_o^{CW} = [1, 1+\frac{n}{2}, 2, 2+\frac{n}{2}, \cdots, \frac{n}{2}-1, n-1, \frac{n}{2}, n] \quad (2)$$

It is notice that, the opposite point cannot be assigned as defined in (2) if n is odd. Since the number of nodes in a graph is odd, a possible and simple way to implement the clockwise opposition is to add an auxiliary node (denoted by "n + 1") to the end of the path and then complete the city count to an even number. Using the definition 2, we can work out the clockwise opposite path of candidate solution and then remove the auxiliary city "n + 1" from the end of opposite path.

On the basis of classical BBO and the definition of opposite path, the original OBBO algorithm was proposed by Ergezer and Simon [28]. The basic procedure of OBBO algorithm is presented as follows in Algorithm 1. It is noticeable that the classical BBO algorithm is enhanced using the scheme of the Opposition-based Differential Evolution [12], namely, opposition-based population initialization (Rows 2–4 in Algorithm 1) and opposition-based generation jumping (Rows 9–13 in Algorithm 1). As far as we know, this exact scheme seems to be a generally-accepted and widely used mode in many opposition-based soft computing algorithms.

Algorithm 1. Original OBBO algorithm.

```
1: procedure OBBO (Problem, Opposition method)
2: Randomly generate initial population, P
3: Generate the opposite of initial population, OP
4: Maintain the fittest amongst P and OP
5: while Generation ≤ gen limit do
6: Perform BBO Migration
7: Remove duplicates from population
8: Calculate the fitness of P
9: if random ≤ Opposition Jumping Rate then
10: Create the opposite population, OP
11: Calculate the fitness of OP
12: Maintain the fittest amongst P and OP
13: end if
14: Restore Elite individuals
15: end while
16: return Best Individual
17: end procedure
```

3. Proposed Algorithm

3.1. Motivation

Simulation results on several symmetric TSP benchmarks in [28] illustrate that OBBO seems to be relatively good at optimal solutions when compared with BBO. However, these experiments expose two serious problems, which may indicate that the initial conclusion is questionable.

Although the terminal condition is set as a constant generation maximum for different comparison algorithms, such as 500 in that paper, the number of candidate solutions explored in a search space by different algorithms may be well-distinguished between each other. It is intuitively obvious that some opposite paths are explored and then considered in each generation for OBBO algorithm, but not for BBO algorithm. Obviously, to compare the performance of BBO and OBBO algorithm, it is unfair and inadvisable scheme, which instead by our termination criterion in Section 4.1.

The other problem, hidden behind OBBO for TSP problems, is that the definition of opposite path is too simple to embody some important characteristics of the candidate path. According to our observations and understanding, the city sequences and the distances between adjacent cities are both the two most core features of a TSP path. It is to be noted clearly that the city sequences here means the relative order in TSP path alone. For example, we only concerned that city 25 is former than city 49, and latter than city 12 in a given TSP path (…, 12, 25, 49, …). However, we do not care about the coordinate of the cities 25, 49 and 12, not to mention the Euclidean distances between them. Furthermore, the Euclidean distance is directly computed based on the geometric coordinates of the nodes of the graph, and yet these TSP paths are differentiated by the sequences when only a graph is given. However, the authors used the former feature (city sequences) and ignored the latter one (distances between adjacent cities) unconsciously or consciously, when they defined the opposite path in [28]. Usually, using the original definition of opposite path presented in [28] will lead inevitably to the low utilization rate of opposite paths as shown in Section 4.2.

It is obvious that the definition of opposite path can be considered as a key to promote the opposition-based soft computing for solving TSP problem. Therefore, an initial motivation of this paper is to further amend the definition of opposite path with the help of the Euclidean distances and city sequences in the graph.

Therefore, the next question is that how to use together with Euclidean distances and city sequences of a candidate path. We think Opposition-Based Learning using the Current Optimum, a significant important variation of OBL in continuous domain, might be a good choice by careful observation of these definitions. As mentioned previously, it was first proposed for function optimization as follows [7].

Definition 3. Let $P = (x_1, x_2, ..., x_D)$ be a point in D-dimensional space, where $x_1, x_2, \ldots, x_D \in \mathbf{R}$ and $x_i \in [a_i, b_i]$, $\forall\ i \in \{1, 2, \ldots, D\}$. The opposite point using the current optimum $\breve{P}_{co} = (\breve{x}_{co1}, \breve{x}_{co2}, \ldots, \breve{x}_{coD})$ is completely defined by its coordinates

$$\breve{x}_{coi} = 2x_{co} - x_i \tag{3}$$

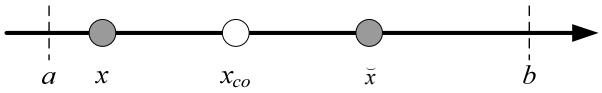
where x_{co} is the optimum solution in the current population.

This definition has similarity, in style, with definition 1 proposed by Tizhoosh, but you will find that the opposite point using the current optimum may be outside the range of valid numbers defined

by [a_i, b_i] if you analyze it carefully. Therefore, the possible solutions include recomputing based on Equation (3) until the new one falls in the range of valid numbers, reproducing a random point, and even using the left or right boundary of valid numbers as an alternative.

Figure 2 illustrates the opposite point using the current optimum $\breve{P}_{co}$ in one dimensional case.

Figure 2. Opposite point using the current optimum defined in domain [a, b]. x is a candidate solution, $\breve{x}$ is the opposite of x and $\breve{x}_{co}$ is its opposite using the current optimum.



The core idea of COOBL may be summarized as that the optimum solution in the current population, replacing the midpoint in a range of variables' current interval, is used as symmetry point of the points and their opposite points. As a result, the opposite points using the current optimum will be in the neighborhood of the global optimum during the process of evolution, especially in the later stage.

In this paper, to redefine the opposite path in discrete domain, the candidate solution and the optimum solution in the current population will be also taken into consideration simultaneity in the similar way.

3.2. Definition of Opposite Path using the Current Optimum

In order to achieve a better solution of TSP efficiently, we modify the definition of opposite path as follows.

As in Figure 3, it is supposed that, n is the number of nodes in a graph and m is the population size. In fact, this figure can decomposed into three parts to comprehend the novel definition of opposite path clearly, which are shown in Figure 4. The first part, the optimal path in the current population P_{co} = [A_1, A_2, …, A_1], as seen in Figure 4a, is translated into line A_1A_4 in Figure 3. In addition, similarly, the candidate path P_i = [B_1, B_2, …, B_1] as the second part of Figure 3 is also translated into line B_1B_3. The clearly common ground between two important parts of Figure 3 is that, they are curves in Figure 4, instead, they are lines in Figure 3. The only reason for the different expression of the same TSP paths is to simplify the most critical figure in this paper. Based on the similar reason, the third part of Figure 3, all cities in the graph as seen in Figure 4c, is ignored in Figure 3. Of cause, you can image it is ubiquity for all cities in this graph in order to understand the following procedure easily.

Figure 3. Novel definition of opposite path (Backward Ellipse).

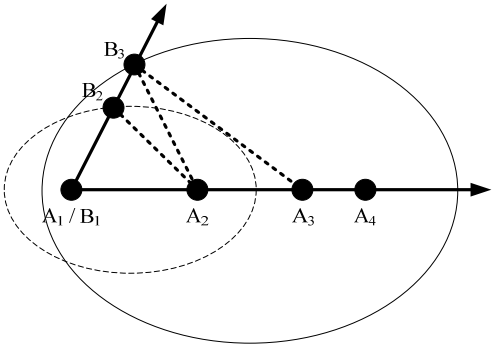
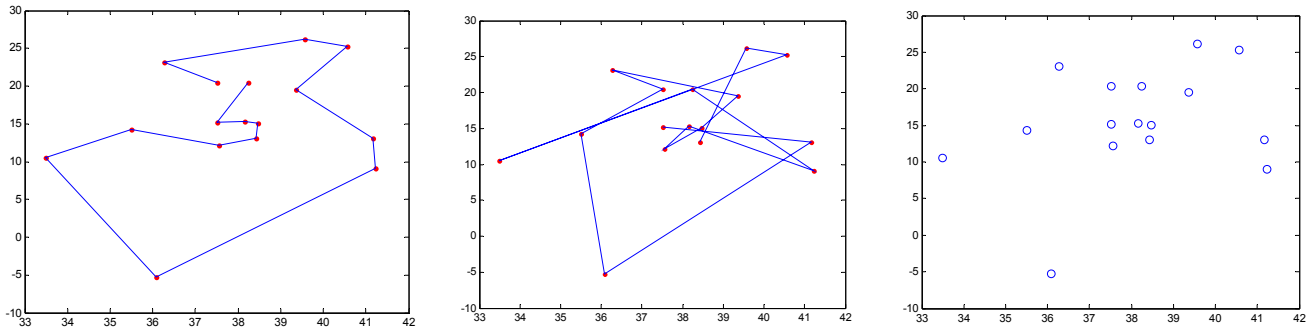


Figure 4. Component elements of opposite path (Backward Ellipse). (**a**) Optimal path in the current population. (**b**) Candidate path. (**c**) Cities in the graph.



(**a**) (**b**) (**c**)

Based on the preliminary and explanation above, the opposite path (Backward Ellipse), P_i^o = [O_1, O_2, …, O_1], of any path P_i = [B_1, B_2, …, B_1] can be defined according to the following procedure.

(1) A set of remaining nodes include all nodes in the graph, and a set of visited nodes is empty in initialization stage.

(2) Let k = 1. The start city, A_1(B_1) of optimal path P_{co} is also determined as the first node of path P_i and its opposite path P_i^o. Then A_1(B_1) is labeled as a visited node and deleted from the set of remaining nodes.

(3) Let $k = k + 1$. An ellipse is determined and denoted by E_k, in which the $(k - 1)$th node and kth node of the optimal path P_{co} are the left focus and the right focus of the ellipse, respectively, and the kth node of P_i is on the boundary of the ellipse E_k.

(4) The kth node O_k of opposite path P_i^o is the nearest node from the set of remaining nodes to the boundary of the ellipse E_k. Then the kth node is labeled as a visited node and deleted from the set of remaining nodes.

(5) Steps 3 and 4 above are iterated until all nodes are included in the set of visited nodes. Then the opposite path, P_i^o, of any path P_i is defined well.

As stated above, the ellipse E_k is determined by means of the $(k - 1)$th node, the kth node of the optimal path P_{co} and the kth node of P_i. The kth node of opposite path P_i^o is close to the boundary of the ellipse E_k. In other words, the kth node of P_i and the kth node of opposite path P_i^o have the same (or similar at least) distance from the $(k - 1)$th node and the kth node of the optimal path P_{co}. Further, the $(k - 1)$th node and the kth node of the optimal path P_{co} can take the place of the whole optimal path P_{co}. Then the path P_i and its corresponding opposite path P_i^o have the same (or similar at least) distance to the whole optimal path P_{co} in the current population in a sense.

Quite similarly, we can define the other two methods, opposite path (Forward Ellipse) and opposite path (Circle), as shown in Figures 5 and 6, respectively. For opposite path (Forward Ellipse), the ellipse E_k is determined by means of the kth node, the $(k + 1)$th node of the optimal path P_{co} and the kth node of P_i. For instance, the second node O_2 of opposite path P_i^o and node B_2 of P_i have the same/similar distance from nodes A_2 and A_3 of the optimal path P_{co}. The third node O_3 of opposite path P_i^o and node B_3 of P_i have the same/similar distance from nodes A_3 and A_4 of the optimal path P_{co}, and so on. For opposite path (Circle), the ellipse E_k is degenerated into a circle C_k, in which the center is the kth node of the optimal path P_{co} and the radius is the distance between the center and the kth node of P_i. For instance, the node B_2 of P_i is located at the boundary of the circle C_2 and the second

node O_2 of opposite path P_i^o is selected around the boundary. The node B_3 is located at the boundary and the node O_3 is selected around the boundary of the circle C_3, and so on.

Figure 5. Novel definition of opposite path (Forward Ellipse).

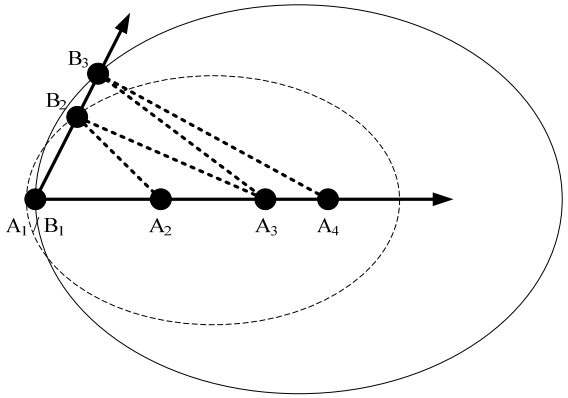
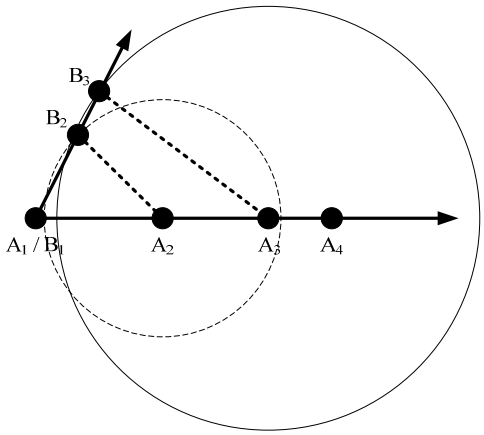


Figure 6. Novel definition of opposite path (Circle).



Without a doubt we introduce the information of the optimum solution in the current population, including the Euclidean distances and the node sequences, into all definitions of opposite path in this paper. The path and its corresponding opposite path have the similar distance to the current optimum in a sense. In addition, the node sequences of opposite path are nearly kept with the original path and optimal path in order, although the direction of opposite path may be slightly different with the original path. In a word, our definition method of opposite path, which is significantly different from [28], considers both the node sequences of candidate paths and the distances between adjacent nodes at the same time. In my opinion, it may be a novel and promising attempt to applying the opposition-based soft computing in discrete domain.

3.3. Our Optimization Method

The novel optimization method, Oppositional Biogeography-Based Optimization using the Current Optimum, for TSP problem suffers the same basic produce like OBBO as shown in Algorithm 1. Actually, the only difference between original OBBO and COOBBO proposed in this paper consists in that the definition of opposite path, which directly influences the population initialization (Row 3 in

Algorithm 1) and generation jumping (Row 10 in Algorithm 1). In the implementation of the algorithm, we should employ the opposite path (Backward Ellipse, Forward Ellipse, or Circle) using the current optimum given in Section 3.2, replacing the opposite path proposed in [28].

4. Experimental Results and Discussions

4.1. Experimental Setup

Here we introduce a random version of OBBO (called ROBBO in this paper) firstly. On the basis of original OBBO, all parts of the proposed algorithm are kept untouched. In addition, instead of using opposite paths for the population initialization and the generation jumping, the random paths, which are generated uniformly in entire search space, are employed in ROBBO algorithm. According to past experience, we admit that this version must be not the best one, yet it is a basic and well-known method, which is firstly used by Rahnamayan in [12].

Benchmark Functions. A comprehensive set of 8 different well-known traveling salesman problems has been employed for performance verification of the proposed approach. All the benchmarks are selected from TSPLIB [33]. The benchmarks sizes vary from small problem to extra large problem to cover a wide range of problem complexity based on category criteria in [32].

Parameter Settings. Note that the main point in this paper is to perform relative comparison among BBO, OBBO, ROBBO and COOBBO. In order to compare the performance without affecting other factors, all conducted experiments use the parameters as follows.

- Population size: 100
- Maximum number of cost function calls: 100,000
- Number of elites: 3
- Opposition jumping rate: 0.3

The termination criterion for all tested algorithms in this paper is that the number of evaluation of cost function is reached the maximum number. It is obvious that, in each iteration of the loop, the number of function evaluation is different for different algorithms due to opposite numbers explored more space. Therefore, the termination criterion in this paper seems more fair and advisable, especially when compared with that used in [28].

Comparison Criteria. We compare the algorithm performance of BBO, OBBO, ROBBO and COOBBO with the help of four well-used criteria in this paper: best solution (BS), computation time (CT), utilization rate of opposite paths (UR), and population diversity (PD). From the definition and design goal, there is no direct relationship among these parameters. Our investigation listed later in this section confirms this conclusion clearly. Thus, we must check all experimental data carefully and discuss them depending on the situation.

First of all, we compare the algorithm performance by measuring the best solution found at the end of the computation, which is the most commonly used metric in literatures. In order to minimize the effect of the stochastic nature of the heuristic algorithms on the measured metric, the reported results for each function is the average over 100 independent Monte-Carlo simulations. It is worth noting that, the data in the following tables are not the minimum value (best result), but rather the mean value of all 100 experiments.

Computation time, also called process time, is the amount of CPU time required for processing instructions of a computer program. The CPU time is usually measured in clock ticks or seconds. It is well known that it depends on a lot of factors such as data structure utilized by the computer program, programming style of programmer or author, and even performance of computer running the algorithm program. However, on the other hand, it also reflects the convergence speed of tested algorithm in a sense.

Utilization rate of opposite paths is a typical comparison criterion for opposition-based soft computing. Refer to early definition proposed in [34], utilization rate of opposite paths is defined as

$$UR = \frac{OP_{res}}{N_p} \tag{4}$$

where OP_{res} is the number of opposite paths reserved as offspring in the next generation, and N_p is the size of the population including all candidate paths and their corresponding opposite paths. A larger UR means more effects of opposite paths during the process of evolution. Furthermore, for different generations, even the same number of candidate solutions (including candidate paths and their opposite paths) may lead to different utilization rate of opposite paths. Generally speaking, the utilization rate of opposite paths is relative high during the former stage, while it is relative low during the later stage. The reason behind this variation is that, the population of candidate solutions is evolved over time towards better solutions and then more and more opposite paths may not be selected into the elitist population.

For population-based intelligent algorithms, population diversity is undoubtedly one of the most important indicators of a population state. Measures to evaluate the population diversity play an important role in adaptively changing search strategies, in analyzing the behavior of soft computing algorithms, and in evaluating individuals to maintain population diversity in a positive manner. In this paper, an edge entropy measure is used to evaluate population diversity. The edge entropy was proposed by Maekawa [35] firstly and defined as

$$H = -\sum_{e \in X} \frac{F(e)}{N_p} \log(\frac{F(e)}{N_p}) \tag{5}$$

where X is a set of edges included in the current population, N_p is the size of the population, and $F(e)$ is the number of edges e in the current population.

4.2. Deeper Analysis of OBBO

In order to intuitively show its drawbacks, the simulation of OBBO algorithm for large scale TSP, ch130, is firstly conducted with the setting listed in Table 1 as in [28]. We show that by using Monte-Carlo method.

The optimal solutions are obtained at the end of 500 generations from both BBO and OBBO. Therefore, the number of candidate solutions explored by BBO is 25000 for each independent Monte-Carlo run. While the mean of candidate solutions explored by OBBO is 32446.5 ± 485.7132 over 200 independent Monte-Carlo simulations. From this viewpoint, it is no surprise that compared with BBO algorithm, the performance of OBBO may be improved as illustrated in [28] for its larger possibility space explored.

The utilization rate of opposite paths is recorded and shown. From Figure 7, we can see that it is decreasing from 50% to 5% sharply over 6 times. After 12 times, it is very close to zero, which means

the opposite paths cannot be selected as elitist solutions and then survived into the next generation. What deserves special mention is that the horizontal ordinate is the number of opposition-based generation jumping, not the number of generation (iteration). The former is almost 30% of the later according to the parameter, opposition jumping rate, used in this simulation.

The utilization rate of opposite paths is recorded and shown. From Figure 7, we can see that it is decreasing from 50% to 5% sharply over 6 times. After 12 times, it is very close to zero, which means the opposite paths cannot be selected as elitist solutions and then survived into the next generation. What deserves special mention is that the horizontal ordinate is the number of opposition-based generation jumping, not the number of generation (iteration). The former is almost 30% of the later according to the parameter, opposition jumping rate, used in this simulation.

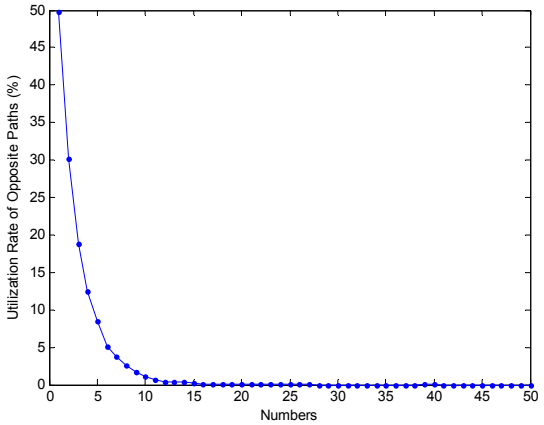
The utilization rate of opposite paths is recorded and shown. From Figure 7, we can see that it is decreasing from 50% to 5% sharply over 6 times. After 12 times, it is very close to zero, which means the opposite paths cannot be selected as elitist solutions and then survived into the next generation. What deserves special mention is that the horizontal ordinate is the number of opposition-based generation jumping, not the number of generation (iteration). The former is almost 30% of the later according to the parameter, opposition jumping rate, used in this simulation.

The utilization rate of opposite paths is recorded and shown. From Figure 7, we can see that it is decreasing from 50% to 5% sharply over 6 times. After 12 times, it is very close to zero, which means the opposite paths cannot be selected as elitist solutions and then survived into the next generation. What deserves special mention is that the horizontal ordinate is the number of opposition-based generation jumping, not the number of generation (iteration). The former is almost 30% of the later according to the parameter, opposition jumping rate, used in this simulation.

Table 1. Simulation settings for TSP problem.

Parameters	Value
Population size	50
Generation limit	500
Number of elites	3
Monte Carlo runs	200
Opposition Jumping Rate	0.3

Figure 7. Utilization rate of opposite paths of OBBO algorithm.



4.3. Experiment Comparison of BBO, OBBO, ROBBO and COOBBO

Results of applying BBO, OBBO, ROBBO and COOBBO to solve 8 benchmark functions are given in Table 2, in which the better results for each case are highlighted in **boldface**.

Table 2. Performance comparison of BBO, OBBO, ROBBO and COOBBO.

Benchmark	Comparison Criteria	BBO	OBBO	COOBBO	ROBBO
berlin52	BS	8604.2732 ± 271.0828	8802.2257 ± 298.9179	**8321.864** ± **243.0838**	8816.3923 ± 310.225
	CT	16.5043 ± **0.033254**	12.8949 ± 0.14883	26.0636 ± 0.40613	**12.7935** ± 0.15288
	UR	-	1.39 ± 0.15792	**17.7021** ± 1.5387	0.98298 ± **0.13792**
	PD	30.9672 ± **0.2238**	32.0102 ± 0.30895	**23.4455** ± 1.7748	31.8362 ± 0.29218
bier127	BS	224444.6936 ± **7873.8039**	247079.3802 ± 8659.568	**186999.418** ± 11339.7602	247017.627 ± 8644.1022
	CT	36.2248 ± **0.18711**	28.0967 ± 0.32596	77.5242 ± 1.7217	**27.9017** ± 0.35213
	UR	-	0.79193 ± 0.12519	**10.0663** ± 0.94305	0.73474 ± **0.11112**
	PD	**37.4125** ± **0.46407**	40.0844 ± 0.74797	37.8905 ± 1.7826	39.9542 ± 0.77365
ch130	BS	14160.151 ± **556.0656**	16163.1156 ± 571.0329	**10735.6857** ± 800.823	16110.5935 ± 580.3632
	CT	36.8155 ± **0.041028**	28.6327 ± 0.28465	80.0928 ± 1.9244	**28.4332** ± 0.28853
	UR	-	0.79987 ± 0.11519	**11.046** ± 1.133	0.72923 ± **0.10295**
	PD	**37.108** ± **0.45369**	39.7573 ± 0.65399	38.2925 ± 1.7171	39.6187 ± 0.6568
kroA150	BS	76274.6154 ± **3082.4458**	86542.4877 ± 3231.4578	**54089.9955** ± 4138.1095	87258.7988 ± 3590.0721
	CT	42.0813 ± **0.039008**	32.634 ± 0.43444	97.9435 ± 2.3961	**32.4983** ± 0.39877
	UR	-	0.84709 ± **0.11022**	**11.5362** ± 1.14	0.71816 ± 0.11195
	PD	**38.4352** ± **0.49502**	41.6914 ± 0.89549	43.8533 ± 2.3142	41.3216 ± 0.78642
kroA200	BS	121550.466 ± **3586.8735**	136496.701 ± 4039.3472	**79433.7261** ± 5793.0264	136436.9172 ± 4140.0182
	CT	55.3055 ± **0.060966**	42.9461 ± 0.60204	146.7581 ± 3.6105	**42.702** ± 0.50824
	UR	-	0.77342 ± 0.1136	**9.8554** ± 0.92318	0.68909 ± **0.10748**
	PD	**40.8049** ± **0.5737**	45.0263 ± 1.3084	51.4226 ± 2.5032	44.635 ± 1.2792
kroC100	BS	38808.7544 ± **1553.4899**	43909.6763 ± 2211.9766	**28786.3067** ± 2089.1848	44319.1631 ± 2033.1415
	CT	28.9781 ± **0.035912**	22.5343 ± 0.2427	56.7792 ± 1.2044	**22.4417** ± 0.25442
	UR	-	0.98634 ± 0.14493	**15.161** ± 1.4206	0.75694 ± **0.10583**
	PD	**35.8608** ± **0.34445**	38.3319 ± 0.6626	36.5919 ± 2.2458	37.8325 ± 0.59057
lin105	BS	28658.7352 ± **1455.8463**	32433.6636 ± 1562.2038	**20653.0943** ± 1528.6458	32414.3953 ± 1534.0982
	CT	30.2712 ± **0.037719**	23.576 ± 0.28172	60.3434 ± 1.2413	**23.4203** ± 0.28112
	UR	-	0.94954 ± **0.11861**	**14.1988** ± 1.4196	0.75923 ± 0.11967
	PD	**36.3301** ± **0.34626**	38.8883 ± 0.60563	38.1673 ± 2.0066	38.4409 ± 0.56284
lin318	BS	293943.2531 ± 6647.7388	324536.2956 ± 5997.1132	**179974.0689** ± 11519.3106	325160.5952 ± **5921.0519**
	CT	86.5313 ± **0.095259**	67.0055 ± 0.77115	298.6361 ± 8.3882	**66.7332** ± 0.83632
	UR	-	0.72979 ± **0.088012**	**7.5774** ± 0.70079	0.66951 ± 0.11773
	PD	**45.9775** ± **0.93707**	52.315 ± 1.9485	67.1539 ± 3.9866	51.8151 ± 1.8752

In order to eliminate the influence of subjective and objective conditions, large scale independent experiments are implemented for each algorithm in this paper. As we all know, the smaller the variance of results is, the more stable the algorithm performance is. From Table 2, we see that BBO has the smallest variance for almost all experiment results, which may clearly indicate that BBO is reliability and robustness.

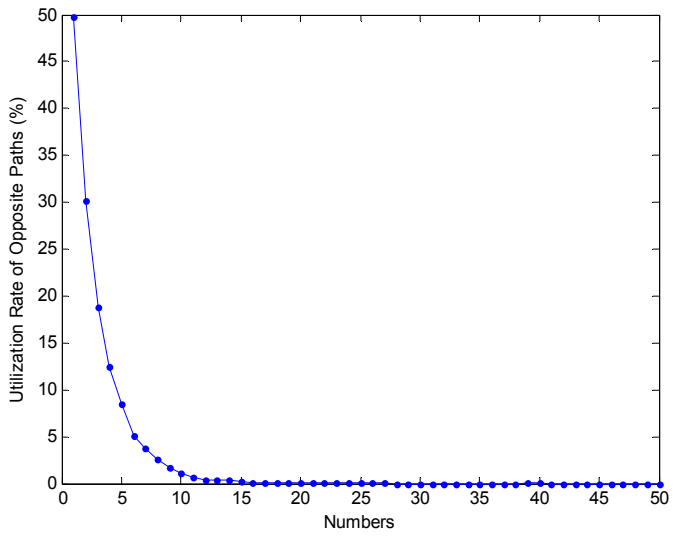
Based on the simulation results in Table 2, COOBBO proposed in this paper achieved the best solution among all. When compared with BBO, OBBO and ROBBO, the mean of the best solution by COOBBO algorithm is lessened, on average, almost by 35.78%, 51.27% and 51.60%, respectively.

It is important to note that the overall performance of OBBO is getting worse than BBO, which may seem contradictory with the conclusion in [28]. In fact, they are not contradictory. A rational reason explained this phenomenon is the termination criterion is pretty different in the two papers. As described above, the two algorithms are not iterated until the fixed iteration is satisfied in [28], which means that the OBBO algorithm will explore more candidate solutions compared with BBO algorithm. However the same algorithms are stopped when the number of evaluation of cost function is reached the maximum number in this paper. In this case, it is not surprising then that OBBO's performance is worse than BBO.

From our point of view, the performance difference of two modified algorithm, OBBO and COOBBO, is attributed to the disparate definition of opposite path. In [28], the opposite of each city is furthermost in the circle. Therefore, the difference of candidate path and its corresponding opposite path is merely style. However, the definition of opposite path proposed in this paper is differs appreciably from that method. The sequence of candidate paths and the distances between adjacent nodes in the tour are considered simultaneously.

As another direct result of the differences in definition of opposite path in the two papers, the utilization rate of opposite paths in this paper is significantly higher than it in [28]. As seen from Table 2, the new method is over 13 times more effective than the traditional one (12.14 *vs.* 0.91 in average). Different from Figure 7 above, the utilization rate of opposite paths in this paper decreases from 50% to 5% over 20 times as seen in Figure 8. Although the decline trend is inevitable, the utilization rate of opposite paths are expected to decrease more slowly over time. Furthermore, as for the great difference between Figures 7 and 8, the utilization rate is no longer decline, after reaching minimal value, such as 4.93% in Figure 8. What some readers will find more surprising is that it wanders and increases slowly with the number of opposition-based generation jumping. That is to say, more and more opposite paths can be survived and further being improved by our proposed algorithm. Using the definition of opposite path from [28], such an event would have been almost inconceivable.

Figure 8. Utilization rate of opposite paths of COOBBO algorithm.



Except for the berlin52 problem, BBO has the lowest edge entropy, which indicates clearly its population diversity is the highest among all tested algorithms. Now, there is an interesting and unexpected phenomenon to us. When compared with COOBBO algorithm, BBO has both higher population diversity and worse solution. In accordance with a common view in the field of population-based search algorithms, the higher population diversity should lead to better solutions of many combinatorial problems. In fact, in order to be successful, a search algorithm needs to establish a good ratio between exploration, which is the process of visiting entirely new regions, and exploitation, which is the process of visiting those regions within the neighborhood of previously visited points [36]. If we think that BBO, to a certain degree, has already a good balance between exploration and exploitation, the combination of opposition-based learning and BBO will break the inherent stability inevitably. Several opposite paths are created based on the original paths and the optimal path in the current population, and then their search space are limited within the neighborhood of previously visited path. Therefore, in my opinion, the advantage of opposition-based learning is the in-depth exploitation, not extensive exploration. Furthermore, the termination criterion in this paper is that the evaluation number is reached the maximum number. Compared with the classical BBO, the evolution process of COOBBO tends to last shorter, because more candidate paths are tried in each iteration. In a word, the original balance between exploitation and exploration in BBO is destroyed by opposition-based learning, and then COOBBO suffers more exploitation, which leads to better solution convergence, and less exploration, which leads to lower population diversity, in the evolution process. According to our understanding, it is the key reason that the experiment results shown in Table 2 are not consistent with the common view in the field of population-based search algorithms.

4.4. Experiment Comparison of OBBO and ROBBO

In order to state clearly the contribution of opposite paths, ROBBO algorithm is introduced in this paper. Unlike the previously defined opposite paths, uniformly generated random paths are employed as opposite paths. As shown from Table 3, the experiment results of OBBO and ROBBO are rearranged and analyzed with software Statistical Product and Service Solutions (SPSS) 14.0 version (SPSS Inc., Chicago, IL, USA) and significance is assumed at 0.05.

As seen from Table 3, we observe that their mainly difference is characterized by three aspects, which are computation time, utilization rate of opposite paths and population diversity. When compared with ROBBO, OBBO has higher utilization rate of opposite paths and worse population diversity, in the statistical sense. According to the definition of opposite path in [28], it is created based on the original path in a limited and controlled manner, though its object is to maximize the proximity between the adjacent vertices. At this time, there has been an appreciable increase in the probability of the opposite path selected as offspring in next iteration. In contrast, the opposite path in ROBBO proposed in this paper is sampled by uniformly distribution over the whole search space, and then ROBBO has better population diversity inevitably.

Seen from the evaluation criterion of best solution found, no significant differences ($p > 0.05$) are observed between OBBO and ROBBO. In other words, the opposite path from [28] has the same contribution with random generated paths. Such being the case, it is futile to attempt to apply the

opposite path from [28], although it is a most plausible story. Maybe the ROBBO algorithm is a good choice to instead OBBO. It is another interesting and valuable conclusion in this paper.

Table 3. Performance comparison of COOBBO and ROBBO.

Benchmark	BS	CT	UR	PD
berlin52	0.743 (8802.2257 *vs.* 8816.3923)	0.000 (12.8949 *vs.* 12.7935)	0.000 (1.39 *vs.* 0.98298)	0.000 (32.0102 *vs.* 31.8362)
bier127	0.960 (247079.3802 *vs.* 247017.627)	0.000 (28.0967 *vs.* 27.9017)	0.001 (0.79193 *vs.* 0.73474)	0.228 (40.0844 *vs.* 39.9542)
ch130	0.520 (16163.1156 *vs.* 16110.5935)	0.000 (28.6327 *vs.* 28.4332)	0.000 (0.79987 *vs.* 0.72923)	0.137 (39.7573 *vs.* 39.6187)
kroA150	0.140 (86542.4877 *vs.* 87258.7988)	0.022 (32.634 *vs.* 32.4983)	0.000 (0.84709 *vs.* 0.71816)	0.002 (41.6914 *vs.* 41.3216)
kroA200	0.918 (136496.701 *vs.* 136436.9172)	0.002 (42.9461 *vs.* 42.702)	0.000 (0.77342 *vs.* 0.68909)	0.034 (45.0263 *vs.* 44.635)
kroC100	0.174 (43909.6763 *vs.* 44319.1631)	0.009 (22.5343 *vs.* 22.4417)	0.000 (0.98634 *vs.* 0.75694)	0.000 (38.3319 *vs.* 37.8325)
lin105	0.930 (32433.6636 *vs.* 32414.3953)	0.000 (23.576 *vs.* 23.4203)	0.000 (0.94954 *vs.* 0.75923)	0.000 (38.8883 *vs.* 38.4409)
lin318	0.460 (324536.2956 *vs.* 325160.5952)	0.018 (67.0055 *vs.* 66.7332)	0.000 (0.72979 *vs.* 0.66951)	0.066 (52.315 *vs.* 51.8151)

4.5. Experiment Comparison of Different COOBBO

As stated above, three methods to define the opposite paths are proposed in this paper, which are called opposite path (Backward Ellipse), opposite path (Forward Ellipse) and opposite path (Circle), respectively. In Section 4.2, opposite path (Backward Ellipse) is employed into COOBBO, and then it is compared with BBO, OBBO and ROBBO. As shown in Table 2, it is demonstrated that the proposed COOBBO algorithm can achieve near optimal performance in terms of best solution found and utilization rate of opposite paths. In order to design better opposite path, three different definitions proposed in this paper are compared in details. Experiment results of applying different version of COOBBO to solve those benchmark functions above are given in Table 4, in which the better results for each case are also highlighted in **boldface**.

From Table 4, we can see that each algorithm has own unique advantages in different comparison criteria over other methods. For instance, COOBBO (Backward Ellipse) is the best algorithm on best solution found and utilization rate of opposite paths, COOBBO (Forward Ellipse) is the best one on population diversity, and COOBBO (Circle) is the best one on computation time.

We believe that, the most likely reason for this is that different definition of opposite path is embedded into classical BBO algorithm. Here let me take opposite node O_2 of node B_2 for example. As seen from Figures 3, 5 and 6, the search process for the opposite node O_2 is limited different search space, such as the ellipse determined by nodes A_1, A_2, and B_2 for Backward Ellipse method, the ellipse determined by nodes A_2, A_3, and B_2 for Forward Ellipse method, and the circle determined by nodes A_2 and B_2 for Circle method, respectively. Hereby we can infer that, if a backward line (such as A_1B_2) is considered to create the opposite path, the exploitation ability can be improved greatly, and the

better utilization rate of opposite paths and overall performance can be achieved. From another perspective, if forward line (such as A_3B_2) is considered in that process, the exploration ability and population diversity can be improved greatly.

Table 4. Performance comparison of different COOBBO.

Benchmark	Optimal Solution	Comparison Criteria	COOBBO (Backward Ellipse)	COOBBO (Forward Ellipse)	COOBBO (Circle)
berlin52	7542	BS	8321.864 ± 243.0838	**8305.1157** ± 241.5853	8333.591 ± 283.9502
		CT	26.0636 ± 0.40613	26.2338 ± 0.43054	**24.4901** ± 0.3307
		UR	17.7021 ± 1.5387	**19.4787** ± 1.7205	11.9418 ± 1.032
		PD	23.4455 ± 1.7748	**20.365** ± 0.99143	31.0124 ± 0.58744
bier127	118282	BS	**186999.418** ± 11339.7602	200390.2364 ± 9909.0866	190145.1356 ± 9264.6665
		CT	77.5242 ± 1.7217	77.871 ± 1.8559	**70.1001** ± 1.2754
		UR	**10.0663** ± 0.94305	9.4902 ± 0.93118	5.5513 ± 0.38539
		PD	37.8905 ± 1.7826	**34.7061** ± 1.3295	47.5324 ± 1.3702
ch130	6110	BS	**10735.6857** ± 800.823	12301.3115 ± 665.8969	11071.1291 ± 602.8048
		CT	80.0928 ± 1.9244	80.2126 ± 1.7268	**72.6642** ± 1.5963
		UR	**11.046** ± 1.133	10.2205 ± 0.96441	6.2528 ± 0.46867
		PD	38.2925 ± 1.7171	**33.9795** ± 1.2991	52.0144 ± 1.1943
kroA150	26524	BS	**54089.9955** ± 4138.1095	62213.9641 ± 3367.083	56154.8562 ± 3199.3826
		CT	97.9435 ± 2.3961	97.909 ± 2.6651	**87.9529** ± 1.6593
		UR	**11.5362** ± 1.14	10.7622 ± 0.91919	6.9752 ± 0.44009
		PD	43.8533 ± 2.3142	**39.1721** ± 1.9173	61.6256 ± 1.7903
kroA200	29368	BS	**79433.7261** ± 5793.0264	95434.8002 ± 4848.5489	93386.4811 ± 3698.4416
		CT	146.7581 ± 3.6105	148.7677 ± 3.9271	**130.3292** ± 3.1375
		UR	**9.8554** ± 0.92318	9.195 ± 0.88397	5.8667 ± 0.3423
		PD	51.4226 ± 2.5032	**46.2781** ± 2.5567	74.8238 ± 3.0705
kroC100	20749	BS	28786.3067 ± 2089.1848	32228.8304 ± 1988.7409	**28124.5048** ± 1469.195
		CT	56.7792 ± 1.2044	57.3461 ± 1.2305	**52.1812** ± 0.91162
		UR	**15.161** ± 1.4206	14.8064 ± 1.5202	10.6852 ± 0.91019
		PD	36.5919 ± 2.2458	**31.6739** ± 1.2889	48.9681 ± 1.039
lin105	14379	BS	**20653.0943** ± 1528.6458	22392.0575 ± 1610.5752	20786.2342 ± 1077.5567
		CT	60.3434 ± 1.2413	61.4 ± 1.2684	**55.1161** ± 1.0321
		UR	**14.1988** ± 1.4196	13.8791 ± 1.2581	10.4896 ± 0.95054
		PD	38.1673 ± 2.0066	**34.3843** ± 1.4297	50.6824 ± 1.088
lin318	42029	BS	**179974.0689** ± 11519.3106	209648.546 ± 10977.459	241710.2118 ± 9377.1761
		CT	298.6361 ± 8.3882	305.7854 ± 8.8387	**261.5351** ± 7.5882
		UR	**7.5774** ± 0.70079	7.1332 ± 0.68193	4.073 ± 0.28234
		PD	67.1539 ± 3.9866	**60.4984** ± 3.586	97.8536 ± 5.4552

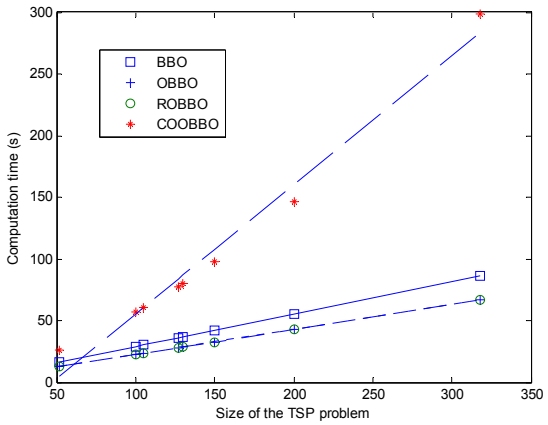
It is obvious that the method of finding opposite path (Circle) has the characteristics of simple process flow, less calculation and high velocity. Comparatively speaking, the overall performance of COOBBO (Circle) algorithm is worst on utilization rate of opposite paths and population diversity when solving those TSP problems. From this fact we may deduce that adjacent nodes in the optimal path in the current population are very necessary for each successful algorithm. The difficult issue

facing us all is how to establish a good ratio between backward adjacent node and forward adjacent node, and it is an important research effort in near future.

4.6. More Discussion on Computation Time

Generally speaking, intelligent heuristic algorithms are more time consuming than other algorithms. On the other hand, computation time of heuristic algorithms normally depend on some measure of problem size, such as the number of cities in TSP problem. As you can see from Figure 9, CPU time consumed by BBO algorithm is increasing linearly and this conclusion is equally applicable to OBBO and ROBBO algorithms.

Figure 9. Comparison of computation time of BBO, OBBO, ROBBO and COOBBO algorithms.



Eight sets of computation time and sizes of TSP problem are carried out by least square method. The fitting results of all tested algorithms are given in Table 5. Herein relative error in the third row is the mean of differences between fitted value and experiment result and it can be defined as following

$$Relative\ error = \sum_{i=1}^{num} \frac{\left| CT_{fit}^{i} - CT_{exp}^{i} \right|}{CT_{exp}^{i}} \tag{6}$$

where CT_{exp}^{i} and CT_{fit}^{i} are the experiment result and fitted value for the *i*th benchmark problem, and *num* is the number of benchmark problems.

Table 5. Fitting results of BBO, OBBO, ROBBO and COOBBO algorithms.

Algorithm	Fitting line	Relative error (%)
BBO	$0.26351x + 2.6561$	0.26629
OBBO	$0.20365x + 2.2008$	0.21228
ROBBO	$0.20297x + 2.1265$	0.21109
COOBBO	$1.04751x - 49.2515$	15.4708

It can be seen from Table 5 that the relative fitting errors of BBO, OBBO and ROBBO algorithms are maintained at moderate rate around 0.2% to 0.3%.

At the same time, it is unfortunate that the computation time of our proposed algorithm is a bit different from the three mentioned above. As can be seen from Figure 9, its computation time is

increased with expedition from 26s for 52 cities to 298s for 318 cities. If we fitted the experiment results by the same method, its relative error is around 15%, which is close to 60 times more than other algorithms. Therefore, we are not sure that the linear fitting is still effective to COOBBO algorithm, and then we surmise that it may be increased exponentially with the increase of the size of the TSP problem. Of course, the inference should be proved or falsified by more experiments in the future. Typically, the main reason for so much time of the COOBBO algorithm is the calculation of Euclidean distance between cities in the produce of forming opposite path.

5. Conclusions

In this paper, a novel oppositional BBO algorithm, called COOBBO, is introduced to solve TSP problems. The proposed algorithm is tested on eight benchmark problems and is found to outperform the standard BBO and OBBO. Further test results indicate that the excellent performance is attributed to the distinct definition of opposite path. The sequence of candidate paths and the distances between adjacent nodes in the tour are considered simultaneously, which may lead to the higher utilization rate of opposite paths.

Population diversity refers to differences among individuals and is used to delimit between exploration and exploitation. However, we are very confused that the proposed algorithm has both better solution and lower population diversity. We think that opposition-based learning is good at in-depth exploitation, not extensive exploration. Therefore, the original balance of COOBBO is destroyed, in which the exploitation process is enhanced and the exploration process is weaken in some cases.

Numerical results demonstrate that there is no significant difference in solution accuracy between OBBO and ROBBO. Considering other factors such as population diversity and computation time, we have abundant proof for thinking that the ROBBO algorithm is a good choice to instead OBBO.

In this paper, we have made a preliminary investigation to define the opposite path and apply it for solving TSP problems. However, it is undeniable that there are many possible definitions and applications of opposite path. By comparing different version of COOBBO, we think that each successful improved algorithm needs to establish a good ratio between backward adjacent node and forward adjacent node. It is an important research effort for us in near future.

Based on the experimental evidence, one of the principal disadvantages of our algorithm is too much long time required to perform it. Without doubt, it is intolerable that each modified algorithm waste so much time on the insignificant improvement. Therefore, our top task in the near future is to increase the efficiency of the algorithm to achieve the best balance between computation time and performance.

It should be emphasized again that the proposed scheme of opposite path is an effective and general method, which can be embedded inside many soft computing algorithms, to improve its performance. Since the key point of this paper is the novel definition of opposite path, we will incorporate that idea into other soft computing approaches, including the state-of-the-art methods, to ascertain what the impact of the opposite path idea. In addition, a comparison to other widely recognized techniques is also required in order to ascertain the utility of the proposed technique.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Nos. 61304082 and 61305083), Shaanxi Province Natural Science Foundation (No. 2012JQ8052) and the Project of Department of Education Science Research of Shaanxi Province (Grant No. 11JK0918).

Author Contributions

Qingzheng Xu and Na Wang designed the research and wrote the paper, Qingzheng Xu and Lemeng Guo performed the simulation experiments, Lemeng Guo and Yongjian He collected and analyzed the data.

Conflicts of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

1. Tizhoosh, H.R.; Ventresca, M. *Studies in Computational Intelligence: Oppositional Concepts in Computational Intelligence*; Springer-Verlag: Berlin, Germany, 2008.
2. Tizhoosh, H.R. Opposition-based learning: A new scheme for machine intelligence. In Proceedings of International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vienna, Austria, 28–30 November 2005; pp. 695–701.
3. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Quasi-oppositional differential evolution. In Proceedings of IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 2229–2236.
4. Ergezer, M.; Simon, D.; Du, D.W. Oppositional biogeography-based optimization. In Proceedings of IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11–14 October 2009; pp. 1009–1014.
5. Rahnamayan, S.; Wang, G.G. Center-based sampling for population-based algorithms. In Proceedings of IEEE Congress on Evolutionary Computation, Trondheim, Norway, 18–21 May 2009; pp. 933–938.
6. Wang, H.; Wu, Z.J.; Liu, Y.; Wang, J.; Jiang, D.Z.; Chen, L.L. Space transformation search: A new evolutionary technique. In Proceedings of ACM/SIGEVO Summit on Genetic and Evolutionary Computation, Shanghai, China, 12–14 June 2009; pp. 537–544.
7. Xu, Q.Z.; Wang, L.; He, B.M.; Wang, N. Opposition-based differential evolution using the current optimum for function optimization. *J. Appl. Sci.* **2011**, *29*, 308–315. (In Chinese)
8. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition versus randomness in soft computing techniques. *Appl. Soft Comput.* **2008**, *8*, 906–918.
9. Rahnamayan, S.; Wang, G.G.; Ventresca, M. An intuitive distance-based explanation of opposition-based sampling. *Appl. Soft Comput.* **2012**, *12*, 2828–2839.
10. Seif, Z.; Ahmadi, M.B. Opposition versus randomness in binary spaces. *Appl. Soft Comput.* **2015**, *27*, 28–37.

11. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition-based differential evolution algorithms. In Proceedings of IEEE Congress on Evolutionary Computation, Vancouver, Canada, 16–21 July 2006; pp. 2010–2017.
12. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition-based differential evolution. *IEEE Trans. Evolut. Comput.* **2008**, *12*, 64–79.
13. Han, L.; He, X.S. A novel opposition-based particle swarm optimization for noisy problems. In Proceedings of International Conference on Natural Computation, Haikou, China, 24–27 August 2007; pp. 624–629.
14. Omran, M.G.H.; Al-Sharhan, S. Using opposition-based learning to improve the performance of particle swarm optimization. In Proceedings of IEEE Swarm Intelligence Symposium, St. Louis, MO, USA, 21–23 September 2008; pp. 1–6.
15. Kaucic, M. A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization. *J. Glob. Optim*. **2013**, *55*, 165–188.
16. Shokri, M. Knowledge of opposite actions for reinforcement learning. *Appl. Soft Comput*. **2011**, *11*, 4097–4109.
17. Ergezer, M.; Sikder, I. Survey of oppositional algorithms. In Proceedings of International Conference on Computer and Information Technology, Dhaka, Bangladesh, 22–24 December 2011; pp. 623–628.
18. Ventresca, M.; Tizhoosh, H.R. Improving the convergence of backpropagation by opposite transfer functions. In Proceedings of International Joint Conference on Neural Networks, Vancouver, Canada, 16–21 July 2006; pp. 4777–4784.
19. Ventresca, M.; Tizhoosh, H.R. Improving gradient-based learning algorithms for large scale feedforward networks. In Proceedings of International Joint Conference on Neural Networks, Atlanta, USA, 14–19 June 2009; pp. 3212–3219.
20. Gao, X.Z.; Wang, X.; Ovaska, S.J. A hybrid harmony search method based on OBL. In Proceedings of IEEE International Conference on Computational Science and Engineering, Hong Kong, China, 11–13 December 2010; pp. 140–145.
21. Qin, A.K.; Forbes, F. Dynamic regional harmony search with opposition and local learning. In Proceedings of 13th Annual Conference on Genetic and Evolutionary Computation, Dublin, Ireland, 12–16 July 2011; pp. 53–54.
22. Malisia, A.R.; Tizhoosh, H.R. Applying opposition-based ideas to the ant colony system. In Proceedings of IEEE Swarm Intelligence Symposium, Honolulu, USA, 1–5 April 2007; pp. 182–189.
23. Banerjee, S.; Tizhoosh, H.R. Visualization of hidden structures in corporate failure prediction using opposite pheromone per node model. In Proceedings of IEEE Congress on Evolutionary Computation, Barcelona, Spain, 18–23 July 2010; pp. 1–5.
24. El-Abd, M. Opposition-based artificial bee colony algorithm. In Proceedings of 13th Annual Conference on Genetic and Evolutionary Computation, Dublin, Ireland, 12–16 July 2011; pp. 109–115.
25. Yang, X.J.; Huang, Z.G. Opposition-based artificial bee colony with dynamic cauchy mutation for function optimization. *Int. J. Adv. Comput. Technol.* **2012**, *4*, 56–62.

26. Xu, Q.Z.; Wang, L.; Wang, N.; Hei, X.H.; Zhao, L. A review of opposition-based learning from 2005 to 2012. *Eng. Appl. Artif. Intell*. **2014**, *29*, 1–12.
27. Ventresca, M.; Tizhoosh, H.R. A diversity maintaining population-based incremental learning algorithm. *Inf. Sci.* **2008**, *178*, 4038–4056.
28. Ergezer, M.; Simon, D. Oppositional biogeography-based optimization for combinatorial problems. In Proceedings of IEEE Congress on Evolutionary Computation, New Orleans, 5–8 June 2011; pp. 1496–1503.
29. Xu, Q.Z.; Guo, L.M.; Wang, N.; Pan, J.; Wang, L. A novel oppositional biogeography-based optimization for combinatorial problems. In Proceedings of International Conference on Natural Computation, Xiamen, China, 19–21 August 2014; pp. 414–420.
30. Simon, D. Biogeography-based optimization. *IEEE Trans. Evolut. Comput.* **2008**, *12*, 702–713.
31. Wikipedia Website, Travelling Salesman Problem. Available online: http://en.wikipedia.org/wiki/Travelling_salesman_problem (accessed on 26 November 2014).
32. Du, D.; Simon, D. Biogeography-based optimization for large scale combinatorial problems. In *Efficiency and Scalability Methods for Computational Intellect*, Igelnik, B.; Zurada, J.M. Eds.; IGI Global: Hershey, PA, USA, 2013; pp. 197–217.
33. Reinelt, G. TSPLIB—A traveling salesman problem library. *ORSA J. Comput.* **1991**, *3*, 376–384.
34. Xu, Q.Z.; Wang, L.; He, B.M.; Wang, N. Modified opposition-based differential evolution for function optimization. *J. Comput. Inf. Syst*. **2011**, *7*, 1582–1591.
35. Maekawa, K.; Mori, N.; Kita, H.; Nishikawa, H. A genetic solution for the traveling salesman problem by means of a thermodynamical selection rule. In Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 20–22 May 1996; pp. 529–534.
36. Crepinsek, M.; Liu, S.H.; Mearnik, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.* **2013**, *45*, 35–50.